IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF NEBRASKA

DAVID PITLOR,

Plaintiff,

vs.

TD AMERITRADE, INC. &
SCHWAB AND CO., INC.,
Defendants.

Case No. 8:23-cv-00407

EXHIBIT 'S':

DECLARATION OF DAVID PITLOR
(28 U.S.C. § 1746)

I, David Pitlor, declare the following:

1. I make this declaration based on my personal knowledge. The statements below summarize device logs, event logs, stack traces, and screenshots I captured in 2018 (collectively, the "materials.")

2. The Barclays 2023 FINRA disciplinary action (AWC. No. 2019061076001) was issued two weeks after the arbitration ended in June 2023. The recordkeeping discrepancies described therein were attributed to "coding errors," which led to my revisiting the logs from the 2018 crashes of the Schwab and TD Ameritrade apps. After considerable time deciphering the logs, I was able to make significant breakthroughs in my understanding, and particularly regarding the Defendants' collusion.

3. I am a licensed professional mechanical engineer. My experience includes the design, operation, and troubleshooting of mechanical, electrical, and digital controls systems.

4. I collected more than 400 screenshots of System logs, Event logs, Stack Traces, and other crash metadata pertaining to the crashes of the Schwab and TD app crashes. This exhibit was

prepared from those materials (and a few other screenshots that document other anomalies pertaining to the Schwab app), without alteration aside from cropping and resizing images. The original files have been or will be made available to Defendant for inspection and copying, including:

a. **Screenshots Nos. 1–278** correspond to Schwab mobile app crashes between February 28 and April 4, 2018.
b. **Screenshots Nos. 279–428**: The TD Ameritrade app crashed on April 26, May 23, May 25, and June 1, 2018. My TD account was supposedly closed in September 2017, but I was able to log in to the "ThinkorSwim" platform up to 2020.

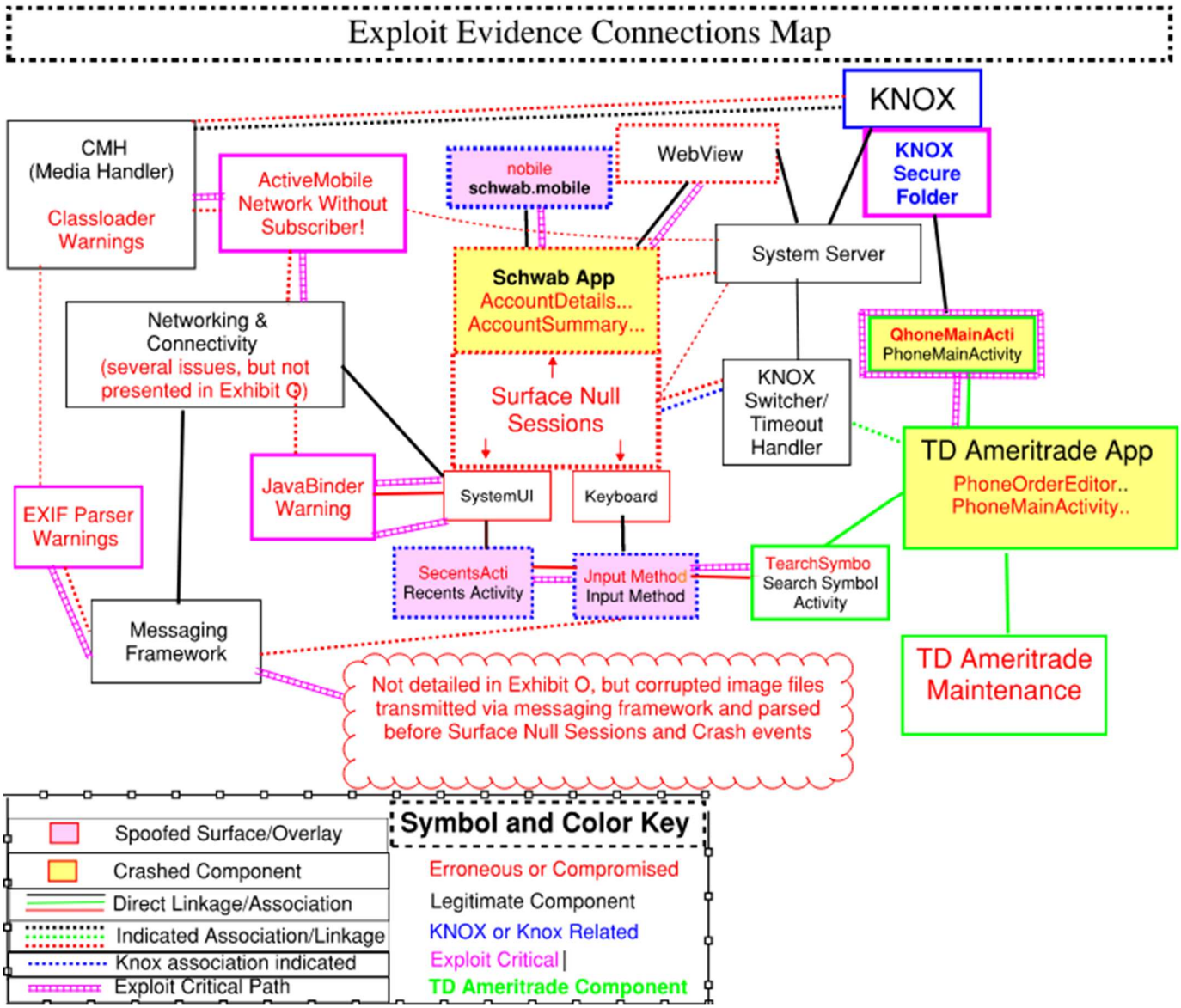
5. Representative examples (¶A-¶R) were selected for this Exhibit that most efficiently demonstrate that my device was compromised by a sophisticated exploit chain that targeted my Schwab account, and moreover that TD Ameritrade was integrally involved in the scheme's execution. I am prepared to present a more detailed analysis of the evidence presented herein and other examples of exploit activities contained within the voluminous body of evidence pertaining to Schwab and TD's "coding errors."

6. A secret user profile was set up on my device that was inaccessible to me (via Samsung Knox). Malicious overlays intruded into the Home (User 0) domain to manipulate onscreen displays, inject and capture inputs, and maliciously interfere with my Schwab app. In addition to monitoring device activity, the exploit enabled redirection of trade orders, selectively, as they were submitted for execution, and thereby enabled transactions to "disappear" from the record. TD Ameritrade's activities concerning spoofed surfaces and Samsung Knox affirms their crucial role in this scheme.

7. Throughout the ensuing analysis, several CVE (Common Vulnerabilities and Exposures) are referenced. While these CVEs describe vulnerabilities that may have been exploited here, an app crash can facilitate similar results by forcing the system to fallback,

temporarily, to a weakened security state, creating a window of opportunity for attackers to escalate privileges and install/operate/teardown the exploit framework. **Accordingly, the CVE citations are provided for context, not attribution, to demonstrate how the systems were vulnerable.**

8. A timeline is presented on the next page to provide context with respect to the relevant events that coincided with the 2018 app crashes, followed by the Table of Contents.



TIMELINE

2018 Schwab and TD Ameritrade Application Crashes

- Feb. 21: Schwab Account Opened
- Feb. 27: first trades occurred:
- Feb. 28: **first Schwab app crashes**
  - account restrictions imposed,
  - $9,999,999.00 added to "Cash on Hold"

- March 1: **Schwab app crash**
  - **TD involvement first indicated w/spoofed surfaces**
- March 3: Schwab app crash
- March 6: **Schwab app crash**
  - $9,999,999.00 removed from "Cash on Hold"
- March 21 **Schwab app crash:**
  - inaccurately documented positions
- March 23 **Schwab app crash** (most profitable day of trading):
  - inaccurately documented positions
- March 26–28:
  - Plaintiff notifies Schwab re: accounting issues and suspected missing funds
- March 30:
  - Schwab Futures Account "closed" without prior notice, removed from account historical data.

- April 2
  - Schwab informed Plaintiff that his Brokerage Account would be closed May (and confirmed Futures account closure)
- April 3: **Schwab app crash**
- April 4: Plaintiff reported to the Fraudfinder and Bridger "Hotlists" on April 4th.
- April 24: **First TD app Crash**
- April 26: Schwab account inaccessible.

- May 2: Scheduled closure date for Schwab account
- May 23: **TD app crash**
- May 25: **TD app crash**

- June 1: **TD app crash**
- June 18: Another entry made to supplement or modify the Fraudfinder and Bridger Hotlists

TABLE OF CONTENTS

A. Stack Traces show “coding errors” impacted Schwab account data 6

B. Schwab’s Equity Awards "coding errors" for login info (but Plaintiff was never enrolled)..... 7

C. Surface Null Sessions and spoofed surfaces compromised critical system components 8

D. Spoofed Surface "Jnput Method" created instead of "Input Method". 9

E. “Image” parsed with over 5412 EXIF components (typical max is a few hundred)..... 9

F. JavaBinder Warnings Indicate Binder UAF exploit 10

G. Network side-channel activated, Exif parsing exceptions, and several additional warnings during media handling--a few seconds before a Schwab app crash..... 11

H. Exploit-related activities--some involving TD Ameritrade--precede Schwab crash..... 12

1. “Passcode disabled due to new User ID” supports malicious “context switches”. 12
2. "QhoneMainActi" spoofs the TD app's "PhoneMainActivity" after Knox User Change..... 12
3. “TearchSymbo”, a hidden spoofed overlay of TD's "SearchSymbolActivity"..... 13
4. "Surface Null Session" involving “Jnput Method” & “TearchSymbo” precedes a crash of the Schwab app crash. 14

I. TD Ameritrade’s “PhoneMainActivity” associates with Knox Secure Folder Task and a Pending Intent..... 15

J. “Illegal State Exceptions” for “PhoneMainActivity” & crashes of the TD app..... 16

K. On March 21, a “Surface Null Session” impacts the Schwab app; the same day as recordkeeping errors that match FINRA AWC No. 2019061076001 (Barclays)……..17

L. A stub WebView package in the Schwab app provided a covert channel for surface overlay attacks, input intercepts/injection, and interference with onscreen rendering. 18

M. A “maintenance” instance of the TD’s app was observed running alongside the standard instance of the TD app while Schwab app crashes occurred 19

N. TD app crash involving “Phone Order Editor Activity”, “Order Editor Parameter Holders” and “Order Entry Fragments.”..... 20

O. JIT inline-cache deoptimization within Zygote facilitated runtime manipulation. 21

P. Dynamic Heap Allocation (DHA) "herding" suppressed security services & reporting..... 22

Q. Schwab app error messages indicate session tampering and cross-context interference 23

R. Other indications that logic and data structures were “imperfectly” rewired by the exploit. . 24

1. “Historical Data is missing from one of more of your accounts,” indicates that transaction histories and balance data had been altered or lost. 24
2. “Personal Value” (Total Account Value) and “Day Change” figures (gain/loss) displayed as “$0.00” and “N/A”, respectively. 24
3. Contradictory info displayed regarding when last transaction occurred supports the other evidence of erroneous dates and manipulated timekeeping 25

A. The Stack traces show that "coding errors" impacted Schwab account data, including "AccountSummaryActivity" and "AccountDetailsTabActivity."

- The exceptions involved "obfuscated" account data structures like com.schwab.mobile.f.a.g and com.schwab.mobile.f.a.z.v()[2]. As such, inside access to Schwab's back-end systems and source code was necessary to "weaponize" the crash events.

No. 180 (March 1)

No. 9 (February 28)

- NullPointerException can compromise an application's stability and data integrity in a variety of ways, including interruption of workflows mid-operation, resource leaks, and state desynchronization.

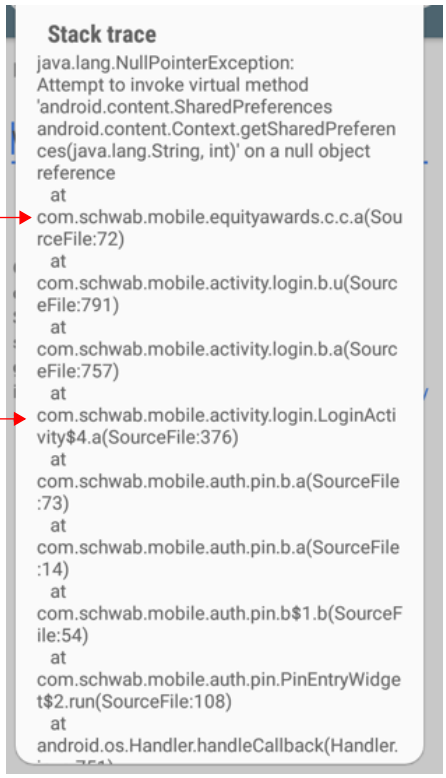
[2] ProGuard is an Android built tool used to optimize and obfuscate app code—renaming to short, meaningless symbols (e.g., com.schwab.mobile.f.a.z.v()). A mapping file is required to convert those symbols into the names of the actual data. In short, this renders implausible any suggestion that Plaintiff, or anybody else without access to Schwab's source code and backend systems, could have been responsible for this exploit.

B. On March 6, 2018, around the time $9,999,999.00 was removed from the *Cash on Hold* balance,[3] Null Pointer Exceptions indicate "coding errors" pertaining to login activities for Schwab's Equity Awards (which Plaintiff was never enrolled in).

- Schwab Equity Awards is Schwab's platform for employees and other plan participants to manage company equity compensation—e.g., viewing grants and vesting schedules, accepting awards, exercising stock options, selling shares, and handling tax withholding/reporting. It's used by companies that hire Schwab as their equity plan administrator. Schwab Equity Awards typically links to a Schwab brokerage account to settle exercises or sales.

- Plaintiff was not enrolled in Schwab Equity Awards, nor was there any legitimate basis for any such activity.

No. 197 (March 6)



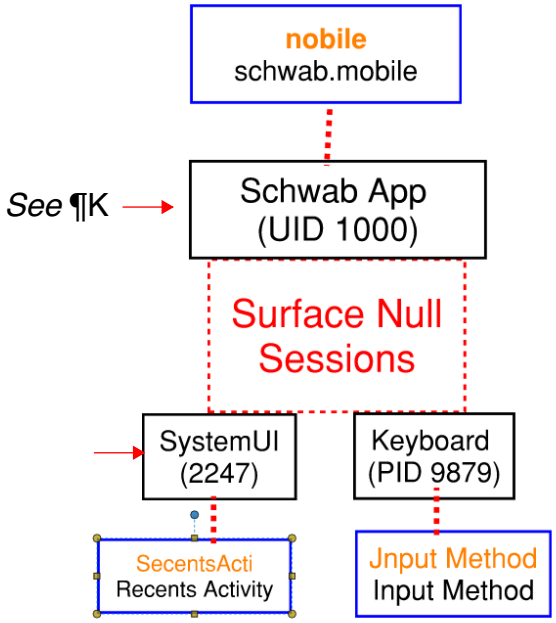
[3] There is substantial evidence that an errant transfer somehow landed in Plaintiff's account on February 28, 2018. Whether it was for $9,999,999.00 or a different amount, the Defendants were keen to remove the funds and eliminate the evidence that any such event occurred. The accounting "sleight of hand" was achieved via a clandestine linkage between Schwab and TD's systems – using Plaintiff's Android smartphone as the intermediary (and utilizing Plaintiff's "closed" TD Ameritrade account as a key facilitator in making funds and transaction data "disappear" from Schwab's domain). After the *Cash on Hold* sum was removed, the contorted account structures continued to be leveraged thereafter to steal money and eliminate the evidence. While the exploit succeeded in some respects (indeed the errors in the official records are rather subtle, making proof of theft virtually impossible without the benefit of the live account balances captured by Plaintiff's screenshots), apparently the app crashes and other errors on the device were significantly louder and messier than the attackers had planned. The noticeable signs of tampering of the Schwab app seem to be unintentional consequences of the exploit's imperfect execution (e.g. "Historical data is missing from some of your accounts", and several other anomalies). It seems the attackers may have been unaware that their "stealthy" exploit was actually rather noisy.

C. Spoofed surfaces and Surface Null Sessions compromised critical system components including the SystemUI[4], Keyboard, and the Schwab app itself.

- The presence of Surface Null Sessions impacting core system processes indicates that privileged Binder[5] connections were rendered vulnerable to compromise.
- On February 28th, the removal of the spoofed surface, "Secents Acti" interfered with the actual "RecentsActivity" and triggerred a Surface Null Session that destabilized the actual systemUI (Process 2247)

No. 75 (System log)

```
02-28 23:34:02.843 I/
SurfaceFlinger( 483): id=2951
Removed SecentsActi (2/7)
02-28 23:34:02.843 I/
SurfaceFlinger( 483): id=2951
Removed SecentsActi (-2/7)
02-28 23:34:02.844 D/Mms/
ComposeMessageFragment( 5010):
Emoticon check
SipHandler.isSipVisible()=true
02-28 23:34:02.844 D/Mms/
ComposeMessageFragment( 5010):
Emoticon check SIP up
02-28 23:34:02.845 D/Mms/
MessageListView( 5010):
getLastReceivedMessage
02-28 23:34:02.846 W/
WindowManager( 1621): Exception
thrown when destroying Window
WindowStateAnimator{b6afc4
com.android.systemui/
com.android.systemui.recents.Rec
entsActivity} surface null session
Session{dec9b1b 2247:u0a10074}:
java.lang.RuntimeException:
Not created this service :
TAG_AOD_WINDOW_MANAGER
```



❖ A Surface Null Session occurs when the window manager attempts to operate on a surface whose session reference is null—commonly because the surface was just torn down, not yet created, its owner process died, **or it belongs to another user context (e.g. Knox).**

❖ Each Surface Null Session was associated with the teardown of a spoofed surface that overlayed a legitimate component.

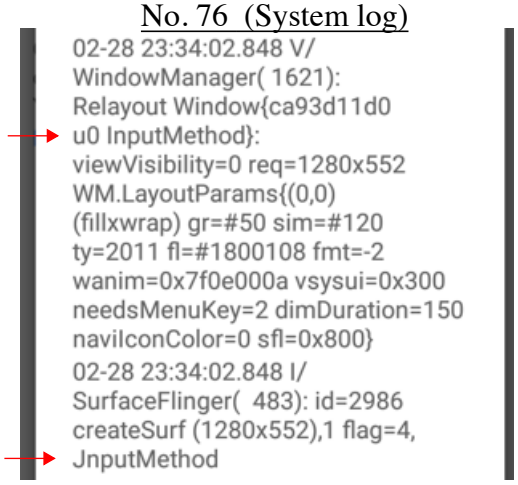
[4] **CVE-2018-9524 (Insufficient SystemUI overlay protections):** This vulnerability permitted a local app to manipulate overlay windows (special UI layers drawn on top of other content) without proper permission checks or bounds validation. This enabled spoofed inputs, hijacking touches/keystrokes, and tricking SystemUI into dispatching privileged actions on the attacker's behalf. Critically, this also facilitated spoofed surfaces (crafted objects masquerading as legitimate display layers) to move across context boundaries (e.g., from Knox into User 0)

[5] **Binders** are Android's inter-prcess communication mechanism that all apps use to communicate with other processes.

D. Immediately after the Surface Null Session on Feb. 28, the system attempts to relayout "Input Method" but the spoofed surface "Jnput Method" was created.

(Not shown: a few milliseconds before Jnput Method was created, secure settings were altered programmatically amidst a flurry of messaging activity– too quick to be user interaction – to alter Input Method settings and Keyboard Shortcuts, apparently leveraging CVE-2021-25393[6] or a similar vulnerability that enabled arbitrary manipulation of Secure Settings.)

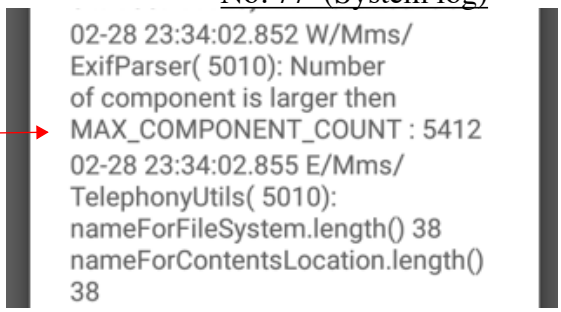
No. 76 (System log)



E. "Image" parsed with abnormally large number of EXIF components

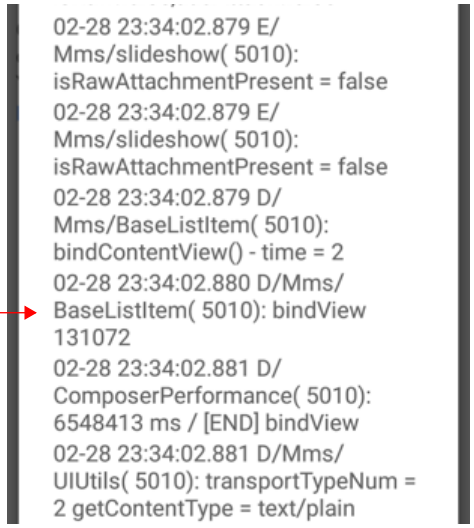
After "Jnput Method" was created, an "image" file with over 5,000+ EXIF components was parsed. Typically, an image file contains dozens to, at most, several hundred components. The context in which this log appears (immediately after Null Session and spoofed surface created), indicates rearming freed Binder references with an attacker controlled object table.

No. 77 (System log)



[6] **CVE-2021-25393** is an improper intent sanitization issue in Samsung's **SecSettings** that permitting arbitrary file read/write with the system UID. This primitive manipulate the settings that govern InputMethod registration. Exploitation of this type of vulnerability was combined with a Binder UAF (CVE-2019-2215 or similar) to reach into the SystemUI and hijack privileged binder paths.

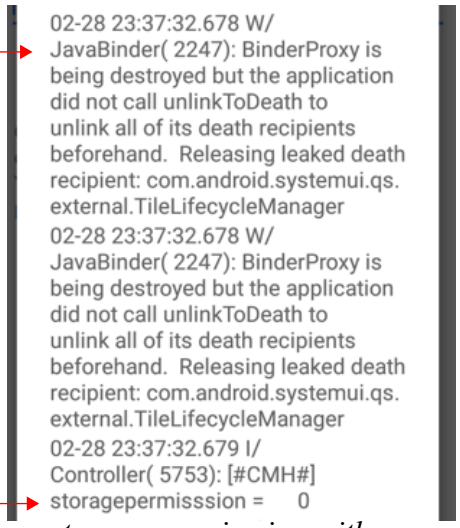
No. 78 (System log)


02-28 23:34:02.879 E/ Mms/slideshow(5010): isRawAttachmentPresent = false
02-28 23:34:02.879 E/ Mms/slideshow(5010): isRawAttachmentPresent = false
02-28 23:34:02.879 D/ Mms/BaseListItem(5010): bindContentView() - time = 2
02-28 23:34:02.880 D/Mms/ BaseListItem(5010): bindView 131072
02-28 23:34:02.881 D/ ComposerPerformance(5010): 6548413 ms / [END] bindView
02-28 23:34:02.881 D/Mms/ UIUtils(5010): transportTypeNum = 2 getContentType = text/plain

❖ Despite the warnings, parsing was not halted and proceeded to bindView. This pattern is consistent with image-parser weaknesses described in **CVE-2019-1987**[7] whereby malformed images were able to drive abnormal allocations.

F. JavaBinder Warnings Indicate Binder UAF Exploit

No. 91 (System log)


02-28 23:37:32.678 W/ JavaBinder(2247): BinderProxy is being destroyed but the application did not call unlinkToDeath to unlink all of its death recipients beforehand. Releasing leaked death recipient: com.android.systemui.qs.external.TileLifecycleManager
02-28 23:37:32.678 W/ JavaBinder(2247): BinderProxy is being destroyed but the application did not call unlinkToDeath to unlink all of its death recipients beforehand. Releasing leaked death recipient: com.android.systemui.qs.external.TileLifecycleManager
02-28 23:37:32.679 I/ Controller(5753): [#CMH#] storagepermisssion = 0

storage permisssion with an extra 's' - uncertain relevance

JavaBinder warnings are logged for the SystemUI process (PID 2247) approximately three minutes after the SystemUI encountered a Surface Null Session (see above ¶C). This sequence is consistent with the exploitation of a Binder Use-After-Free (UAF) vulnerability.[8]

A Binder UAF can be exploited to marshal a malicious parcel (a serialized data container used in Binder transactions for passing parameters across processes). In this instance, a malicious "image" file was inserted into a privileged rendering path (Samsing CMH) while a screenshot was being processed ("piggybacking" on legitimate activity).

The JavaBinder warnings confirm that, after the freed binder introduced the malicious parcel, the dangling binders were cleaned up. Just a few hundred milliseconds later, an abnormal "image" file was parsed (see next page, ¶G), ultimately leading to a crash of the Schwab app.

[7] The decode-then-bindView sequence despite parser irregularities aligns with CVE-2019-1987's description of out-of-bounds writes in Skia's image sampling when parsing malformed images. This is cited here to show the type of weakness implicated.

[8] Binder User-After-Free is a of memory corruption issue in Android's Binder IPC (Inter-process communication) framework where a Binder object is prematurely freed but subsequently referenced, allowing attackers to manipulate the freed memory slot.

G. Network side-channel activated, Exif Parsing Exceptions, and several additional warnings--a few seconds before a Schwab app crash, as screenshot is processed by Samsung CMH (confirmed by preceding screenshots- not shown here).

Binder warning (¶F) logged at 23:37:32.678

No. 87

System log

02-28 23:37:32.396 D/ MountService(1621): getExternalStorageMountMode : final mountMode=1, uid : 5004, packageName : com.samsung.cmh

02-28 23:37:32.411 I/ ActivityManager(1621): Start proc 5753:com.samsung.cmh:CMH/ 5004 for service com.samsung.cmh /.service.CMHService

02-28 23:37:32.412 E/Zygote (5753): v2

02-28 23:37:32.412 I/ libpersona(5753): KNOX_SDCARD checking this for 5004

02-28 23:37:32.412 I/ libpersona(5753): KNOX_SDCARD not a persona

02-28 23:37:32.412 D/ CMHServiceInterface(18573): Sending intent to CMH

02-28 23:37:32.413 E/Zygote (5753): accessInfo : 0

02-28 23:37:32.413 I/art (1621): Background partial concurrent mark sweep GC freed 40649(2MB)

No. 89

System log

CompatibilityInfo(1621): applicationScale - 1.0

02-28 23:37:32.482 I/SA (4927): [OCP] Cursor is not null

02-28 23:37:32.487 I/art (5753): Starting a blocking GC AddRemoveAppImageSpace

02-28 23:37:32.488 W/System (5753): ClassLoader referenced unknown path: /system/priv-app/ CMHProvider/lib/arm64

02-28 23:37:32.491 D/ SmartAlertController(1621): SmartAlert - Found Missed Event

02-28 23:37:32.491 D/ SmartAlertController(1621): SmartAlert - already registered or Setting disabled

02-28 23:37:32.496 I/ [SC]CommonUtil(4888): email : 0 tncState :0 nameCheckState : 0 emailValidationState : 0

02-28 23:37:32.503 I/ [SC]MediaDetectionService(4888): action: com.samsung.android.s cloud.MEDIA_SYNC, operation: 1

No. 93

System log

Controller(5753): [#CMH#] setSuccessState = true

02-28 23:37:32.861 I/ Controller(5753): [#CMH#] setSuccessState latch is down

02-28 23:37:32.861 I/ Initializer(5753): [#CMH#] CMH initialization ends

02-28 23:37:32.862 I/ Controller(5753): [#CMH#] com.android.providers.media Insert

02-28 23:37:32.863 I/ ServiceManager(5753): [#CMH#] FaceService already stopped:

02-28 23:37:32.863 I/ ServiceManager(5753): [#CMH#] StoryService already stopped:

02-28 23:37:32.863 I/ Controller(5753): [#CMH#] processInsertAction,No of files to insert: 1,Origin:com.android.providers.media

02-28 23:37:32.869 I/ DayAndMonthTask(5753): [#CMH#] after mutex release

02-28 23:37:32.872 I/art (5803): Starting a blocking GC AddRemoveAppImageSpace

No. 94

System log

[#CMH#] Bound to Package com.samsung.dcmservice

02-28 23:37:32.901 W/ NetworkIdentity(1621): Active mobile network without subscriber!

02-28 23:37:32.901 W/ NetworkIdentity(1621): Active mobile network without subscriber!

02-28 23:37:32.901 W/ NetworkIdentity(1621): Active mobile network without subscriber!

02-28 23:37:32.901 W/ NetworkIdentity(1621): Active mobile network without subscriber!

02-28 23:37:32.901 W/ NetworkIdentity(1621): Active mobile network without subscriber!

No. 95

System log

NetworkIdentity(1621): Active mobile network without subscriber!

02-28 23:37:32.926 I/ ExifInterface_JNI(5753): Raw image not detected

02-28 23:37:32.932 W/ ExifInterface(5753): Invalid image: ExifInterface got an unsupported image format file(ExifInterface supports JPEG and some RAW image formats only) or a corrupted JPEG file to ExifInterface.

02-28 23:37:32.932 W/ ExifInterface(5753): java.io.IOException: Invalid marker: 89

02-28 23:37:32.932 W/ ExifInterface(5753): at android.media.ExifInterface.getJpegAttributes(ExifInterface.java:1835)

No. 38

Event log

sysui_action(1621): [324,false]

02-28 23:37:36.184 I/ sysui_action(1621): [325,88856]

02-28 23:37:36.188 I/ am_finish_activity(1621): [0,174567712,7565,com.android.vending/com.google.android.finsky.activities.AppCrashProxy,app-request]

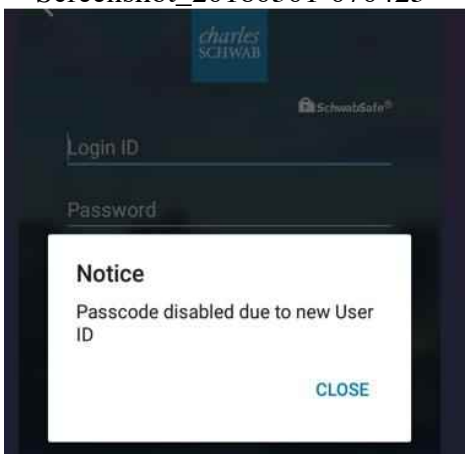
02-28 23:37:36.238 I/ sf_frame_dur(483): [Application Error: com.schwab.mobile, 1,0,0,0,0,0,0]

02-28 23:37:36.339 I/ am_proc_start(1621): [0,5840,10018,com.google.android.gms.ui,activity,com.google.android.gms/.feedback.FeedbackActivity]

❖ The warning "Active mobile network without subscriber" appears to reflect a networking side-channel being activated just seconds before the Schwab app crash on February 28 – coinciding with a maliciously crafted "image" file being parsed

H. The next morning, on March 1, a series of exploit-related activities precede the crash of the Schwab app, some of which involve TD Ameritrade.

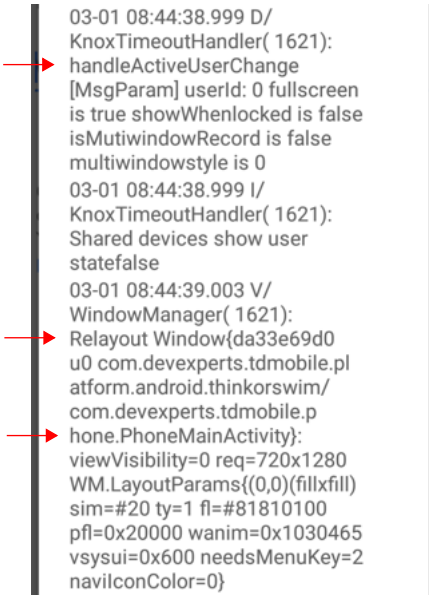
No. 111 (March 1)
Screenshot_20180301-070423



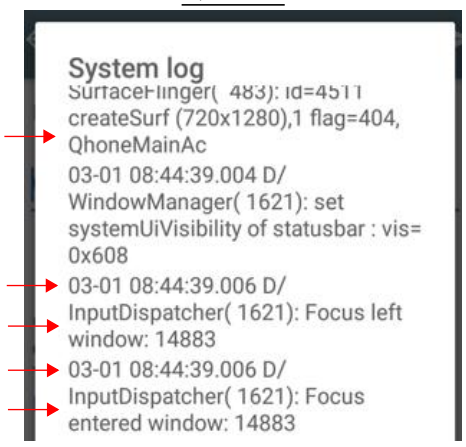
1. **First, at 7:04 am, the Schwab app alerted: "Passcode disabled due to new User ID" (just after another crash event), supporting the notion of "context switches" involving secret users.**

2. **Then, at 8:44 am, the relayout of PhoneMainActivity by the TD Ameritrade app results in the creation of a spoofed surface QhoneMainActi—immediately after Knox handles a user change back to the home User 0.**

No. 150 (System log)


03-01 08:44:38.999 D/ KnoxTimeoutHandler(1621): handleActiveUserChange [MsgParam] userId: 0 fullscreen is true showWhenlocked is false isMutiwindowRecord is false multiwindowstyle is 0
03-01 08:44:38.999 I/ KnoxTimeoutHandler(1621): Shared devices show user statefalse
03-01 08:44:39.003 V/ WindowManager(1621): Relayout Window{da33e69d0 u0 com.devexperts.tdmobile.platform.android.thinkorswim/ com.devexperts.tdmobile.phone.PhoneMainActivity}: viewVisibility=0 req=720x1280 WM.LayoutParams{(0,0)(fillxfill) sim=#20 ty=1 fl=#81810100 pfl=0x20000 wanim=0x1030465 vsysui=0x600 needsMenuKey=2 naviIconColor=0}

No. 151


System log
createSurf (720x1280),1 flag=404, QhoneMainAc
03-01 08:44:39.004 D/ WindowManager(1621): set systemUiVisibility of statusbar : vis= 0x608
03-01 08:44:39.006 D/ InputDispatcher(1621): Focus left window: 14883
03-01 08:44:39.006 D/ InputDispatcher(1621): Focus entered window: 14883

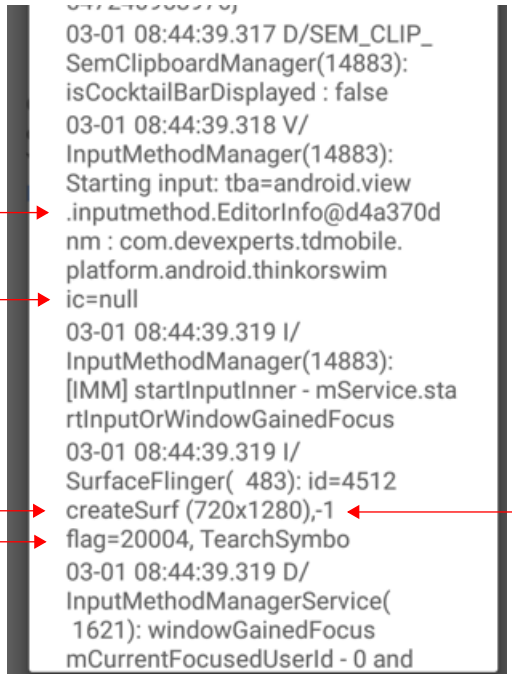
- (Right Screenshot **No. 151):** Instantaneous focus leaving and entering the same window, 14883, indicates a context switch (i.e. user change). Window 14883 belongs to the TD Ameritrade app (as shown on next page). This is strong indication that a Knox user had crossed into the User 0 space (Knox is supposed to be a completely isolated user context and should not interact with apps in User 0 space).

3. **A hidden overlay surface was created, "TearchSymbo" TD Ameritrade process (14883).**

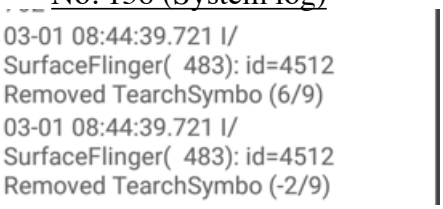
- "TearchSymbo" is a spoof of TD Ameritrade's "SearchSymbolActivity."

- "Tearch Symbo" was removed less than a half-second after it was created. During that brief timespan, the spoofed surface JnputMethod was removed – triggering a Surface Null Session for the Keyboard process (as shown on next page).

No. 155 (System log)



No. 158 (System log)



- The display parameters for "TearchSymbo" indicate that **this spoofed surface was hidden – invisible -** not a normal UI component of the app.

  - **-1** → indicates no explicit parent layer (created at top-level or default parent). A legitimate sub-view or input field would be attached to the app's surface, but the '-1' parameter means this surface was created stand-alone at root level—classic malicious overlay behavior.

  - **flag=20004** → Internal bitmask of surface creation flags. 0x20000 typically indicates HIDDEN or special property. 0x00004 often means SECURE (surface content not allowed in screenshots/screen recording). Combined → 0x20004 → a hidden/secure or offscreen surface.

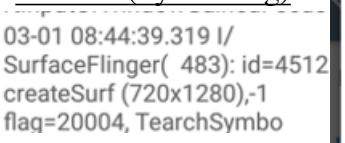
❖ TearchSymbo" fleeting presence is consistent with exploit scaffolding rather than normal application behavior. There's no user interaction with it, no touch, no view hierarchy. This implies low-level injection: TearchSymbo was designed to infiltrate, execute as specific task, and leave as few traces behind as possible.

4. **A Surface Null Session coincides with the teardown of "Jnput Method" amidst activity involving "TearchSymbo" and TD app processes—during a sequence that ultimately culminates with a Schwab app crash.**

- This Surface Null Session occurred while TearchSymbo was present. Then, WindowManager immediately destroys the actual "SearchSymbolActivity"—belonging to the TD app.

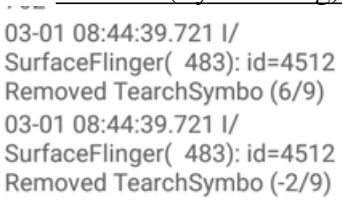
- Jnput Method was created the previous day, coinciding with another Schwab app crash (*See* ¶D)

No. 155 (System log)



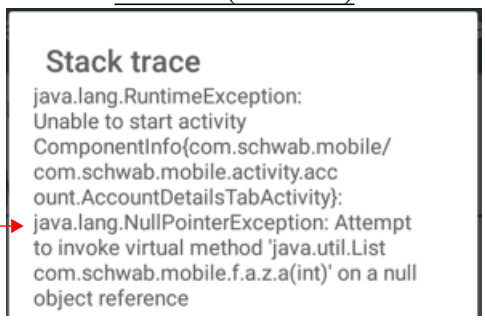
No. 157

No. 158 (System Log)



- The negative refcount (e.g., −2/10) indicates a teardown underflow—SurfaceFlinger attempted to remove a surface that was no longer tracked (its window token had already been invalidated). In other words, the window was torn down after its token became stale, suggesting a duplicate/late removal—**consistent with cross-context interference.**

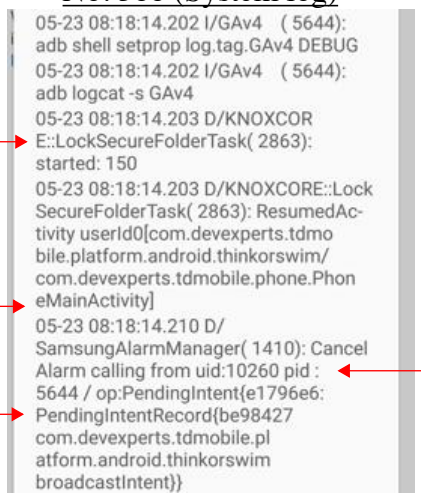
No. 180 (March 1)



❖ The Schwab app crashed due to a "java.lang.RuntimeException"—the same type of exception logged by the Surface Null Session moments earlier. This plausibly supports the notion that the instability caused by the Surface Null Session propagated into a crash of the Schwab app.

I. TD Ameritrade's "PhoneMainActivity" associates with Knox Secure Folder Task and a Pending Intent (Broadcast Intent).

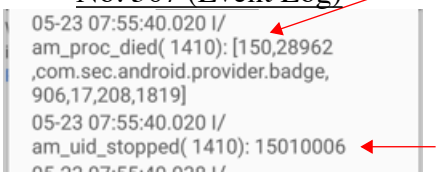
No. 388 (System log)


05-23 08:18:14.202 I/GAv4 (5644): adb shell setprop log.tag.GAv4 DEBUG
05-23 08:18:14.202 I/GAv4 (5644): adb logcat -s GAv4
05-23 08:18:14.203 D/KNOXCOR E::LockSecureFolderTask(2863): started: 150
05-23 08:18:14.203 D/KNOXCORE::LockSecureFolderTask(2863): ResumedActivity userId0[com.devexperts.tdmobile.platform.android.thinkorswim/com.devexperts.tdmobile.phone.PhoneMainActivity]
05-23 08:18:14.210 D/SamsungAlarmManager(1410): Cancel Alarm calling from uid:10260 pid : 5644 / op:PendingIntent{e1796e6: PendingIntentRecord{be98427 com.devexperts.tdmobile.platform.android.thinkorswim broadcastIntent}}

- Seconds after a crash of the TD app on May 23, a Secure Folder task was locked, and TD's *PhoneMainActivity* resumed in the home User 0 domain.
- As presented above, PhoneMainActivity was spoofed by QhoneMainActi on March 1. (*See* ¶H.2).

- Multiple instances of Knox user activity appears in the May 23 logs (multiple crashes occurred that day).

No. 367 (Event Log)



No. 358 (Event Log)

05-23 07:49:52.630 I/ am_proc_bound(1410): [0,28273,com.samsung.android.knox.containeragent]
05-23 07:49:52.775 I/am_kill (1410): [0,24994,com.pinsight.dw, 906,17,DHA:empty #33]
05-23 07:49:52.780 I/ am_uid_running(1410): 15001250
05-23 07:49:52.805 I/ am_proc_start(1410): [150,28292,15001250,com.samsung.android.knox.containeragent,broadcast,com.samsung.android.knox.containeragent/.switcher.knoxusage.KnoxUsageReceiver]
05-23 07:49:52.832 I/ am_uid_idle(1410): 10260

❖ Pending Intents allow one app to perform actions on behalf on another's "intent." With escalated privileges, the attackers could abuse this mechanism to delegate actions and bypass standard permission checks via broadcast intents – precisely what could facilitate covert data transfers and/or command cross-context execution between Knox and User 0 boundaries.

- A PendingIntent remains active when the underlying app is not running, has been force-closed and/or removed from memory. It even survives reboots.
- In this case, the Knox Secure Folder is locked, and the Pending (Broadcast) Intent is canceled, certainly consistent with teardown of the exploit framework.

❖ UID 10260 is the TD App, further indication of their direct coordination with Knox Secure Folder activities.

❖ In Android's multi-user model, a Knox/Secure Folder runs as a separate Android user/profile (User 150), giving it its own sandboxed user space distinct from the primary user.

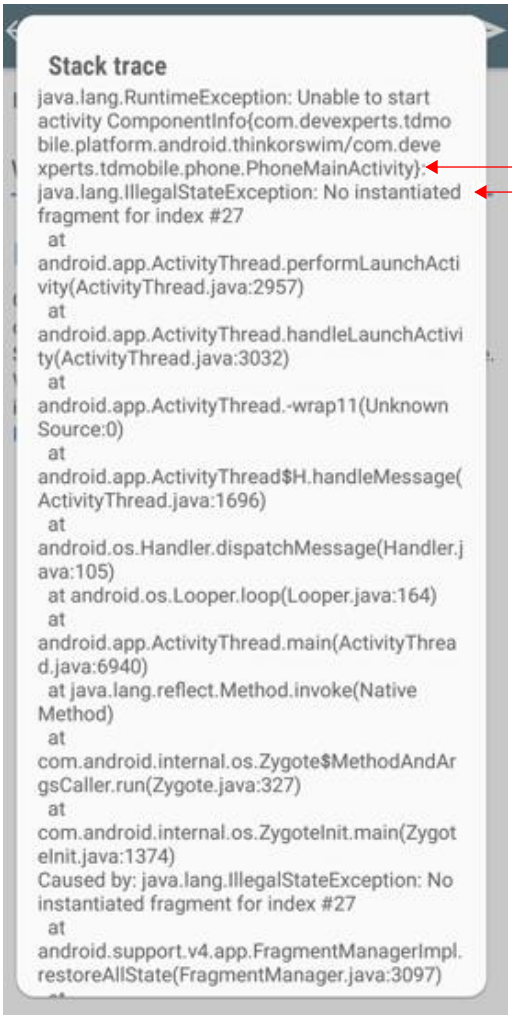
- Apps inside that Knox profile get per-profile UIDs computed as userId + appId, so their UIDs differ from the same app in user 0. So, as seen above in screenshot No. 368, the Knox persona is activated for 15001250, and then the TD app (UID 10260) is immediately idled.

J. "Illegal State Exceptions" for "PhoneMainActivity" are associated with the crashes of the TD app on April 24 and June 1, 2018 and are consistent with teardown (i.e., uninstallation) of the exploit.

- The surrounding logs (including post-crash relaunches and component removals—not shown here), provide additional support that these crashes represent a device-level teardown/reset of components associated with the TD app, rather than a benign exception occurring during a normal UI transition. *(See ¶14 regarding deoptimization of hot call sites on April 24.)*

No. 303 (April 24) - Event Log

No. 420 (June 1)



- An exploit that crosses profiles (Knox ↔ User 0) can bypasses isolation via hooking into system services and altering lifecycle anchors like window tokens and binders. When uninstalled, a stale or mismatched state—such as fragments saved from the other context—can trigger **IllegalStateExceptions** on restore/teardown (e.g., "No instantiated fragment for index"). Indeed, this is consistent with the log evidence here.

K. On March 21, 2018, a "Surface Null Session" is directly associated with the Schwab application; that date coincides with transaction-date mismatches of the type described in FINRA AWC No. 2019061076001 (Barclays Capital).

- The Surface Null Session is logged by the system_server (PID 1623 / UID 1000) for the Schwab mobile app: com.schwab.mobile and is preceded by the removal of a surface named "nobile"—apparently a spoof of the Schwab app itself.
- Knox/Secure Folder log entries appear immediately afterward, essentially handed controls back to the home User space: This pattern is consistent with a teardown of an injected/overlay surface from a Knox (Secure Folder) context that failed to detach cleanly, with system_server (UID 1000) mediating cleanup.
- Knox TimeoutHandler entries appear in close proximity to the other Surface Null Sessions as well (within a few hundred milliseconds), thus reinforcing the notion that the spoofed surfaces are Knox[9] owned but managed to trespass into User 0 space.

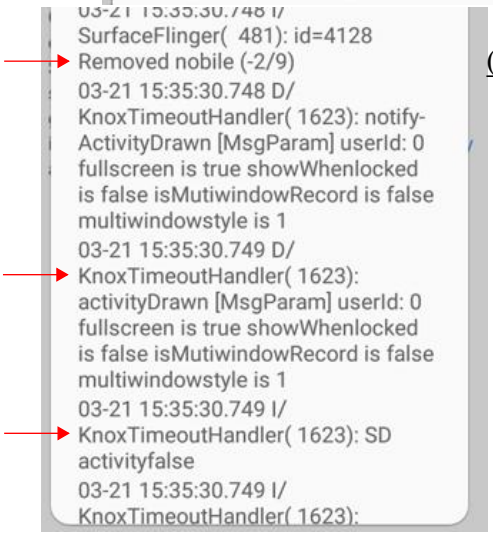
No. 235

Event log
am_crash(1623): [32507,0,com.schwab.mobile, 949501508,java.lang.NullPointerException,Attempt to invoke virtual method 'java.util.List com.schwab.mobile.common.a.x.a(int)' on a null object reference,SourceFile,133]
03-21 15:35:26.718 I/ am_finish_activity(1623): [0,108380737,8860,com.schwab.mobile/.activity.account.AccountDetailsTabActivity,force-crash]
03-21 15:35:26.719 I/ wm_task_moved(1623): [8860,1,7]
03-21 15:35:26.722 I/ am_focused_activity(1623): [0,com.schwab.mobile/.activity.account.AccountSummaryActivity,finishActivity adjustFocus]
03-21 15:35:26.722 I/ am_pause_activity(1623): [0,108380737,com.schwab.mobile/.activity.account.AccountDetailsTabActivity]
03-21 15:35:27.230 I/ am_restart_activity(1623): [0,194723084,8860,com.schwab.mobile/.activi

No. 264 (System log)

SurfaceFlinger(481): id=4128
Removed nobile (2/9)
03-21 15:35:30.740 W/ WindowManager(1623): Exception thrown when destroying Window WindowStateAnimator{dabb14 Starting com.schwab.mobile} surface null session
Session{c67abb3 1623:1000}: java.lang.RuntimeException: Not created this service : TAG_AOD_WINDOW_MANAGER

No. 266 (System log)


SurfaceFlinger(481): id=4128
Removed nobile (-2/9)
03-21 15:35:30.748 D/ KnoxTimeoutHandler(1623): notify-ActivityDrawn [MsgParam] userId: 0 fullscreen is true showWhenlocked is false isMutiwindowRecord is false multiwindowstyle is 1
03-21 15:35:30.749 D/ KnoxTimeoutHandler(1623): activityDrawn [MsgParam] userId: 0 fullscreen is true showWhenlocked is false isMutiwindowRecord is false multiwindowstyle is 1
03-21 15:35:30.749 I/ KnoxTimeoutHandler(1623): SD activityfalse
03-21 15:35:30.749 I/ KnoxTimeoutHandler(1623):

[9] Several Knox vulnerabilities have been acknowledged by Samsung, including those that essentially allow attackers to commandeer Knox.
CVE-2017-10963 – MITM lets an attacker **install any app into the Knox container** (without the user's knowledge), enabling control/data leakage inside the container.
CVE-2019-6744 – **Secure Folder lock-screen** handling flaw enabled local access to Secure Folder contents.
CVE-2024-20856 – **Improper Authentication** lets physical attackers access Secure Folder without proper authorization

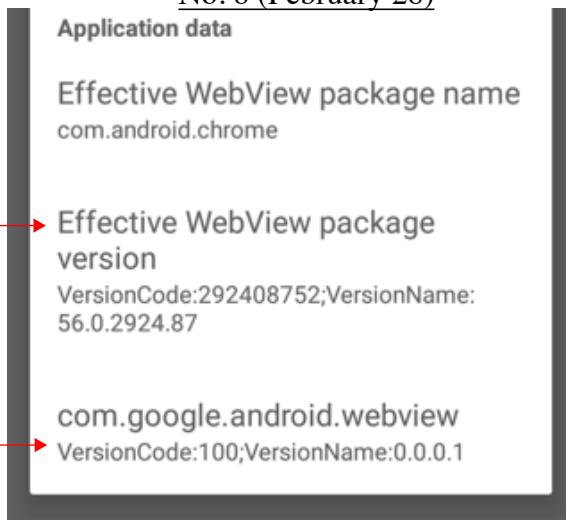
L. A stub Webview package in the Schwab app provided a covert channel for surface overlay attacks, intercept/inject inputs, and interfere with onscreen data rendering.

No. 6 (February 28)

- (right) "System information" affirms that this crash pertains to the Schwab mobile app.

- (below) The "Application data" lists two 'Effective WebView package version' entries, including a placeholder stub (VersionCode 100; VersionName 0.0.0.1). That configuration is not expected in a production build and indicates the app was compiled or reconfigured to use a non-production WebView—which typically requires source-level modification and elevated device privileges.

No. 8 (February 28)

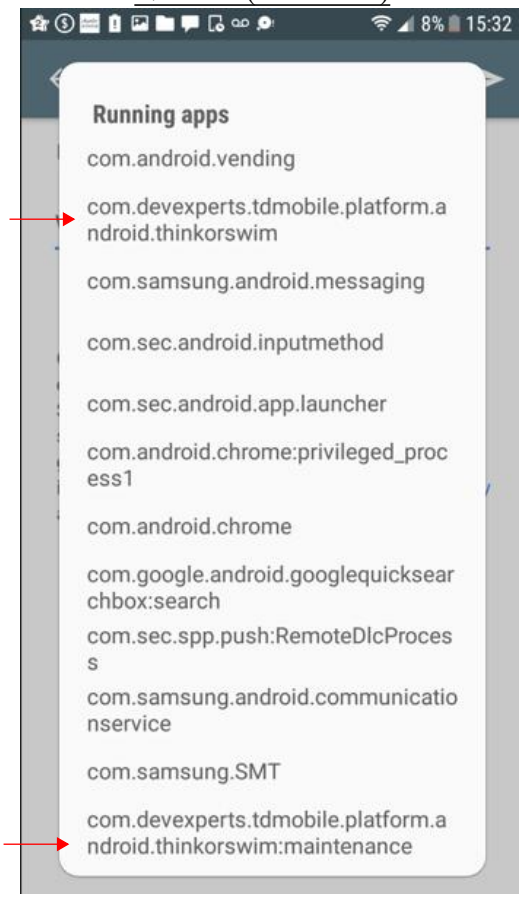


❖ Several instances (not shown here) involving WebView "privileged processes" and "sandboxed processes," including on March 3: the WebView's "privileged_process2" dies and restarts at the start of a Schwab app crash sequence. (Screenshots Nos. 183-193)

❖ Stub WebViews are dangerous because they're placeholders that can be swapped or redirected to an attacker-controlled provider, bypassing the hardened, trusted WebView runtime. An attacker can then load hidden pages and run arbitrary JavaScript in the app's context—stealing cookies/tokens, abusing JavascriptInterface bridges, and manipulating in-app data flows.

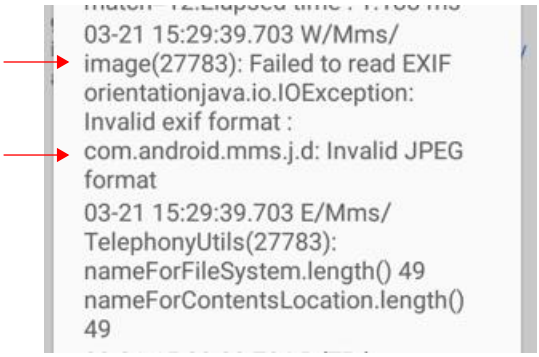
M. A "Maintenance" of the TD's app was observed running alongside the standard instance while Schwab app crashes occurred including on March 21.

No. 205 (March 21)

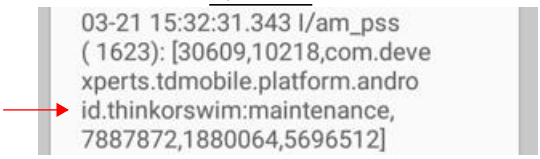


- ThinkorSwim Maintenance was active performing background tasks three minutes after a warning for an Invalid JPEG that processed.

No. 237



No. 229



- And three minutes later, the Schwab app crashed.

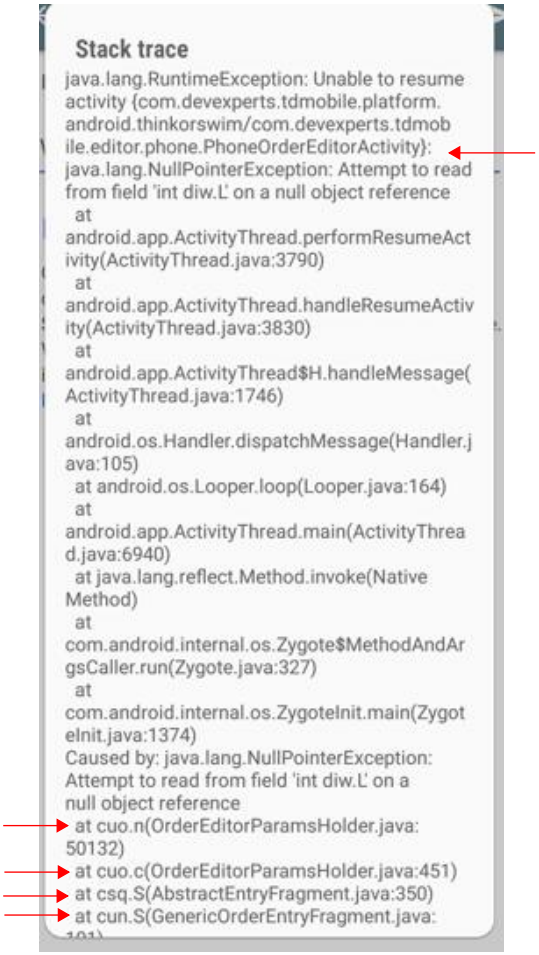
No. 235

❖ The "maintenance" instance of TD Ameritrade may have been a malicious overlay that could communicate with Schwab'sversion of the Schwab app that was tooled to interface with the TD back-end system. The key was to the trick was the overlay of surfaces on the device.

N. On May 23, coding errors cause the TD app to crash with Null Pointer Exceptions involving "Phone Order Editor Activity", "Order Editor Parameter Holders" and "Order Entry Fragments."

- This provides a unique glimpse into nature of the "coding errors": the mechanics of order data handling was altered—absolutely consistent with the issues described by June 2023 FINRA disciplinary action against Barclays, and necessary to facilitate a "rewiring" of the account logic and data structures to interfere and interpose with the Schwab data.

No. 344 (May 23)



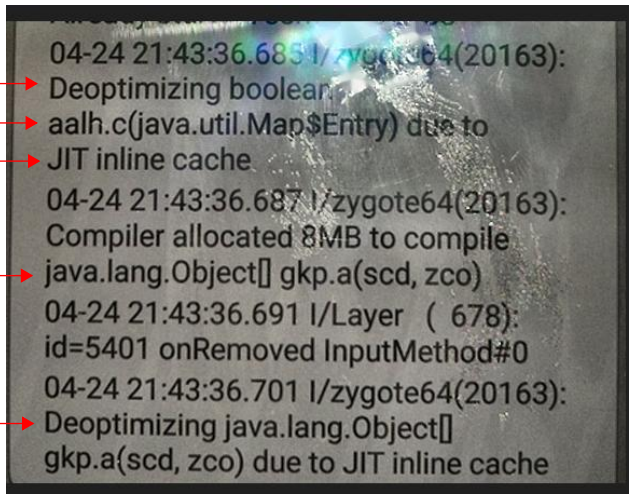
O. JIT inline-cache deoptimization within Zygote provided a means to rewire application logic and data structures.

- "Hot" call sites (i.e., frequently executed code paths) such as *java.lang.Object* and *java.util.Map$Entry* were deoptimized, which enabled malicious runtime manipulations.

  - **java.lang.Object** is the root of the Java type hierarchy. Forcing deoptimization at Object-related call sites opens a massive attack surface—attacker-controlled logic can interpose at the most frequently exercised dispatch points and alter comparisons, lookups, or other operations that affect app integrity.

  - **java.util.Map$Entry** (i.e., Map.Entry): reads/writes and iteration of kev/value pairs — a prime place to intercept lookups/returns of sensitive data (account balances, transaction records).

No. 327



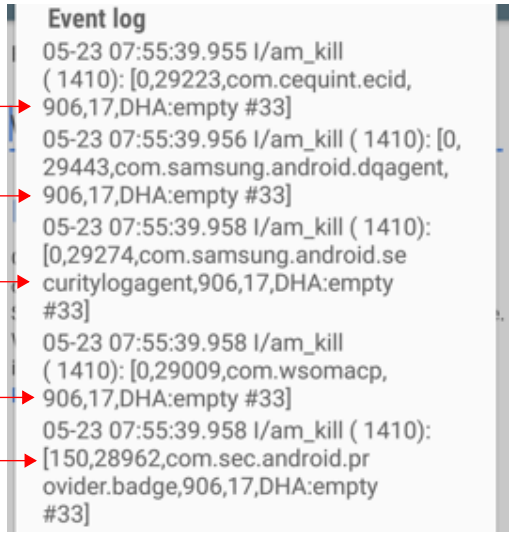
P. Dynamic Heap Allocation (DHA) manipulation ("herding") altered the runtime environment to suppress logging, analytics, and crash reporting.

- Android's Security Log Agent and other core system services were misclassified as cached or empty (rendering them expendable even though they should never be classified as such) and were terminated one minute before a TD app crash event on May 23. (multiple crashes were observed that day, but only a very few visible signs appear in the logs).

No. 366

- ❖ Processes for both User 0 and the Knox UID 150 (Android secure badge provider) were simultaneously targeted for elimination – consistent with teardown of a system-level exploit that violated Knox boundaries.

- ❖ The killed processes are all tagged the same OOM_score_adj of 906 (immediately reclaimable) and herded into DHA slot #33 to be emptied - a tell-tale sign that the allocator was biased.

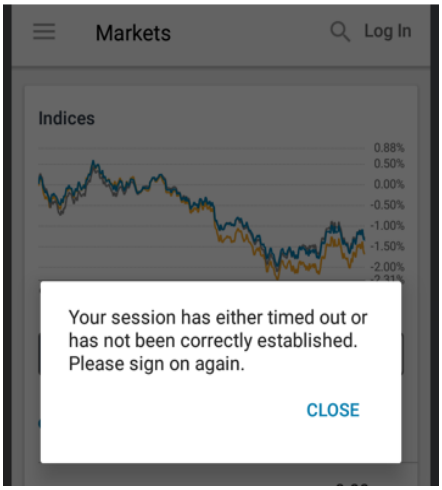

Event log
05-23 07:55:39.955 I/am_kill (1410): [0,29223,com.cequint.ecid, 906,17,DHA:empty #33]
05-23 07:55:39.956 I/am_kill (1410): [0, 29443,com.samsung.android.dqagent, 906,17,DHA:empty #33]
05-23 07:55:39.958 I/am_kill (1410): [0,29274,com.samsung.android.se curitylogagent,906,17,DHA:empty #33]
05-23 07:55:39.958 I/am_kill (1410): [0,29009,com.wsomacp, 906,17,DHA:empty #33]
05-23 07:55:39.958 I/am_kill (1410): [150,28962,com.sec.android.pr ovider.badge,906,17,DHA:empty #33]

- . By suppressing the crash response, the attacker's could take advantage of the devices weakened state (like a guard stepping away from the security booth, permitting intruders to enter uncontested and turn off the security cameras). It is suspected that the app crashes themselves were not intended to be noticeable: Just a little blip, and then the app restarts as it's brought back into focus. The crashes of the Schwab app particularly tended to occur during this restart process. The Attackers may have even believed that everything was executing properly, and for all intents and purposes, it was—from their end, but the chaos on my device was very noticeable. Speculation regarding the root cause of the exploits "imperfections" is beyond the scope of this declaration.

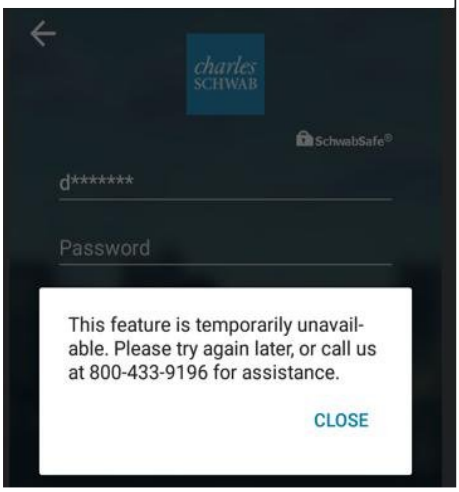
Q. Error messages displayed frequently by the Schwab app are consistent with session tampering and cross-context interference

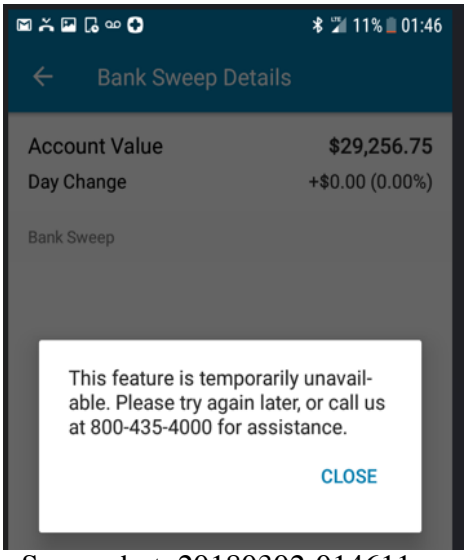
- In addition to the crashes and other accounting discrepancies, the Schwab mobile app repeatedly surfaced modal alerts indicative of session-state resets, user-identity context changes, and data-availability faults, including the following:
  - "Your session has either timed out or not been correctly established,"
  - "Passcode disabled due to new User ID,"
  - "This feature is temporarily unavailable…"
- While error messages are often the result of benign circumstances, the frequency of these occurrences provides additional context that supports the conclusions drawn herein, namely that these are artifacts from multiple users' sessions clashing on the device..

Screenshot_20180302-031144

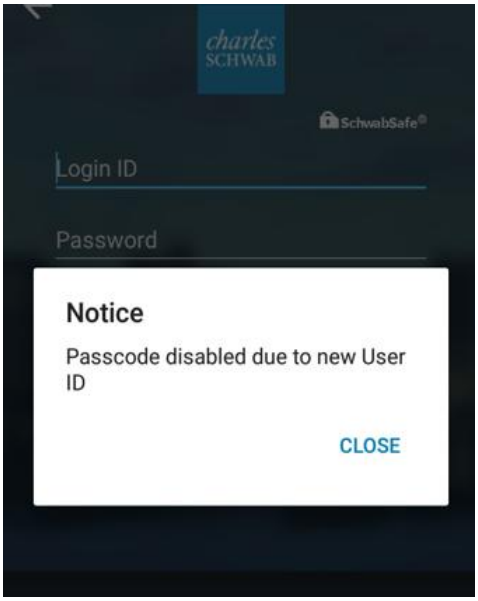

Markets
Log In
Indices
Your session has either timed out or has not been correctly established. Please sign on again.
CLOSE

Screenshot_20180308-013202





Screenshot_20180302-014611



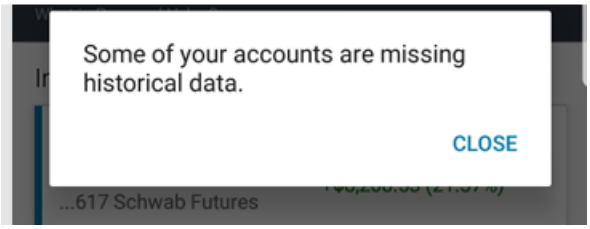
No. 111

R. Other issues with the Schwab app indicate that app logic and data structures were "imperfectly" rewired by the exploit.

While the exploit succeeded in misrepresenting key account figures to conceal funds targeted by theft, there were obvious signs of tampering:

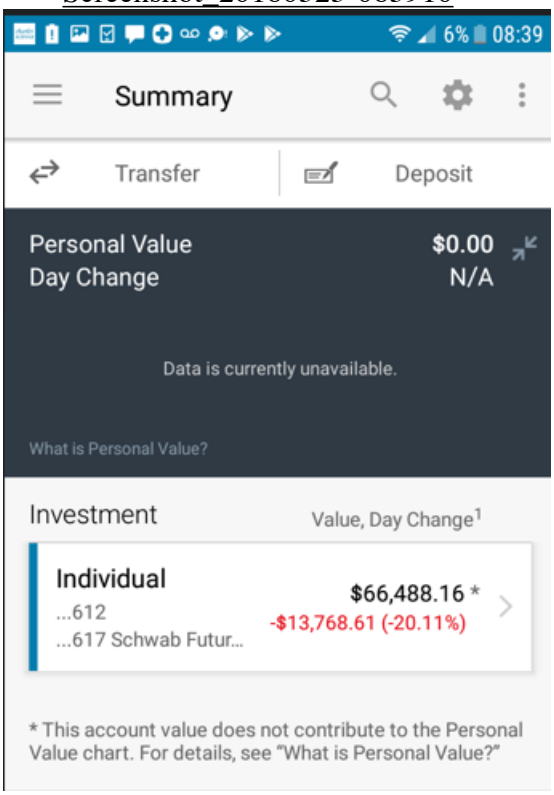
1. **Notifications regularly appeared upon sign-in, alerting that: "Historical Data is missing from one of more of your accounts," a clear indication that transaction histories and balance data had been altered or lost.**

Screenshot_20180323-144454 (and 6 other instances)

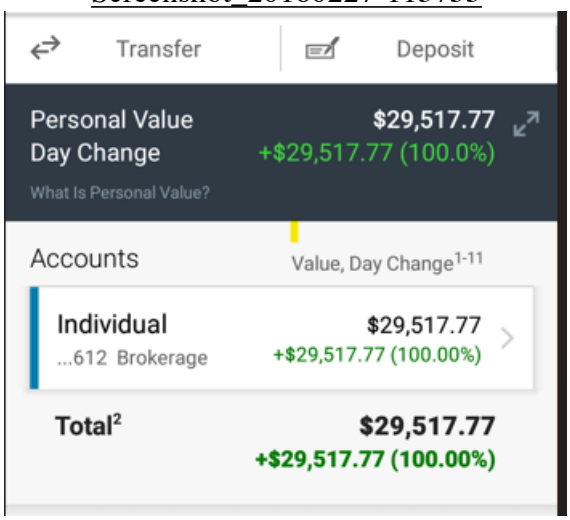


2. **The overall account "Personal Value" and "Day Change" figures (Total Account Value, gain/loss for the day) displayed as "$0.00" and "N/A", respectively.**

Screenshot_20180323-083910



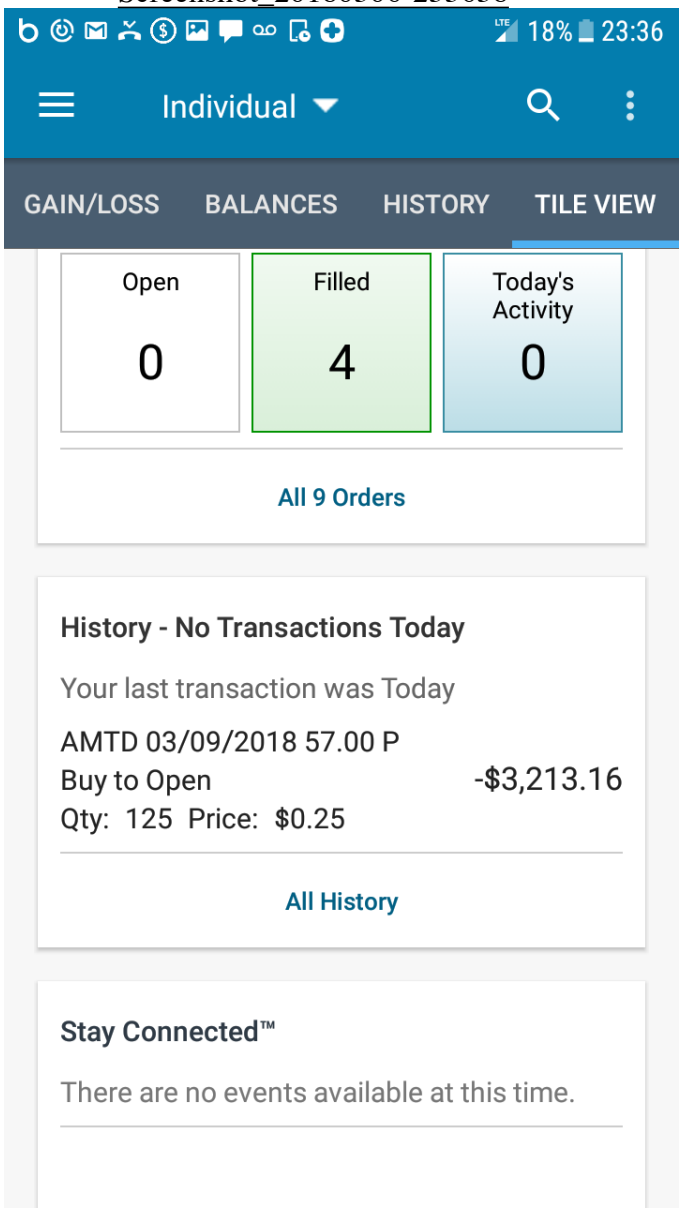
Screenshot_20180227-113755



- ❖ This account does not contribute to the Personal Value chart" further suggests that account balances had been "rewired." particularly considering that, before the Futures Account was opened, there was no such issue.

3. **Inconsistency regarding when the last reported transactions occurred supports the other evidence of erroneous dates and manipulated timekeeping.**

Screenshot_20180306-233638



I declare under penalty of perjury that the foregoing is true and correct.

Executed on November 6, 2025,

Respectfully submitted,

/s/David Pitlor

David Pitlor, P.E.
Licensed Professional Mechanical Engineer
Nebraska Certificate No. E-17959